# Viewing ReLU Networks as Input-Dependent Matrix Selection (WD Form)

## 1. Reordered View

A typical ReLU layer can be written (with slight reordering over the conventional view) as:

$y = W\mathrm{ReLU}(x)$

where ReLU acts elementwise:

- outputs $x_i$ if $x_i>0$
- outputs 0 otherwise

Introduce a **decision (gating) matrix** D(x), defined as a diagonal matrix:

- $D_{ii}(x) = 1$ if $x_i>0$
- $D_{ii}(x) = 0$ if $x_i\leq 0$

Then:

$\mathrm{ReLU}(x) = D(x)x$

So the layer becomes:

$y = WD(x)x$

---

## 2. Synthetic Matrix Interpretation

Define a **synthetic matrix**:

$S(x) = WD(x)$

Then:

$y = S(x)x$

Key point:

- The layer is now a **linear transformation**, but
- The matrix S(x) **depends on the input**

So the network is a **piecewise-linear system**, where each region of input space corresponds to a different matrix.

---

# 3. What Does D(x) Do?

Since D(x) is diagonal with 0/1 entries:

- Multiplying WD(x) **selects columns of W**
- Columns corresponding to inactive neurons are zeroed out

So:

- S(x) = **a submatrix of W with some columns removed (zeroed)**

---

# 4. Regional Interpretation (Piecewise Linear Regions)

Each distinct binary pattern of D(x) defines a **region** in input space:

- A region corresponds to a fixed set of active neurons
- Within that region, D(x) is constant
- Therefore, S(x) is constant

So:

- The network partitions space into regions
- Each region uses a **fixed linear map S**

This is the classical “piecewise linear” view—but here made explicit in matrix form.

---

# 5. “Regional Lookup” Inside W

From this viewpoint:

- W is a **repository of columns**
- D(x) selects which columns are active

- S(x) is assembled by **looking up and combining those columns**

So the computation is:

> "Given input x, select a subset of columns from W, then apply the resulting matrix."

This is a **structured lookup** rather than a fully recomputed transformation.

---

# 6. Hash Table Interpretation

The diagonal entries of D(x) form a **binary vector**:

$d(x) \in \{0,1\}^n$

This can be interpreted as a **hash key**:

- Each input produces a binary pattern (active/inactive neurons)
- That pattern indexes a specific matrix S(x)

So conceptually:

- d(x) = **hash of the input**
- S(x) = **value retrieved from a table**
- Output = S(x)x

Important nuance:

- The "table" is not explicitly stored
- Instead, it is **implicitly encoded in W**
- The lookup is implemented via multiplication with D(x)

---

# 7. Weight Reuse in S(x)

There is **massive reuse of weights**:

- Every column of W is reused across many different S(x)
- Different regions share overlapping subsets of columns

So:

- The number of possible regions (patterns of D) is exponential

- But the number of parameters (in W) is only quadratic

This creates:

### Combinatorial reuse

A small set of weights generates a large number of effective linear maps.

### Structured sharing

Nearby regions differ by only a few bits in D(x), so:

- Their matrices S(x) are highly similar
- Only a few columns change

---

# 8. Significance

## 8.1 Efficiency of Representation

Instead of storing a separate matrix per region:

- The network **factorizes** all matrices into:
    - a shared matrix W
    - a binary selector D(x)

This is a very compact representation of a large family of linear maps.

---

## 8.2 Generalization

Because regions share columns:

- Learning in one region affects many others
- The model generalizes by **recombining learned features**

---

## 8.3 Fast Conditional Computation

The computation:

$S(x) = WD(x)$

is cheap because:

- D(x) is diagonal and binary
- It reduces to masking columns

So the network implements:

> "Conditional linear computation with minimal overhead"

---

### 8.4 Associative Memory View

Each column of W can be viewed as storing a **feature-to-output association**.

Then:

- D(x) selects a subset of features
- The output is the **sum of their associated outputs**

This is analogous to:

- retrieving multiple entries from memory
- combining them linearly

---

# 9. Summary

Rewriting a ReLU layer as:

$y = WD(x)x = S(x)x$

reveals that:

- The network performs **input-dependent matrix selection**
- D(x) acts as a **binary gating / hashing mechanism**
- W is a **shared memory of reusable components**
- S(x) is a **region-specific linear map built by lookup**

This perspective highlights:

- piecewise linear structure
- combinatorial capacity
- efficient weight reuse
- and a natural interpretation as a **hash-indexed associative system**